LAPORAN

RENCANA TUGAS MANDIRI (RTM) KE-IV

MATA KULIAH DATA MINING

“ MEMBUAT KODE SCRIPT ALGORITMA KLASIFIKASI NAÏVE BAYES SECARA SCRATCH MENGGUNAKAN PYTHON “

DISUSUN OLEH:

Sandria Amelia Putri (21083010005)

DOSEN PENGAMPU:

Tresna Maulana Fahrudin S.ST., M.T. (NIP. 199305012022031007)

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”

JAWA TIMUR

2023

- Import terlebih dahulu library yang diperlukan yaitu pandas, numpy, seaborn, dan matplotlib.pyplot. Library tersebut memiliki fungsi di antaranya:
  - Pandas: Manipulasi dan analisis data.
  - Numpy: Melakukan operasi numerik pada data array atau matriks.
  - Seaborn: Membuat visualisasi data yang menarik dan informatif.
  - Matplotlib.pyplot: Membuat visualisasi data dalam bentuk grafik, plot, chart, atau gambar.

```
[1]: # import library
     import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
```

- Membuat dataframe rekomendasi pembelian komputer yang akan digunakan untuk melakukan klasifikasi Naïve Bayes. Dataframe tersebut akan disimpan ke dalam variabel df.

```
[2]: # membuat dataframe rekomendasi pembelian komputer
data = [{'id':1, 'Age':'<=30', 'Income':'High', 'Student':'No', 'Credit_rating':'Fair', 'Class: buys_computer':'No'},
        {'id':2, 'Age':'<=30', 'Income':'High', 'Student':'No', 'Credit_rating':'Excellent', 'Class: buys_computer':'No'},
        {'id':3, 'Age':'31-40', 'Income':'High', 'Student':'No', 'Credit_rating':'Fair', 'Class: buys_computer':'Yes'},
        {'id':4, 'Age':'>40', 'Income':'Medium', 'Student':'No', 'Credit_rating':'Fair', 'Class: buys_computer':'Yes'},
        {'id':5, 'Age':'>40', 'Income':'Low', 'Student':'Yes', 'Credit_rating':'Fair', 'Class: buys_computer':'Yes'},
        {'id':6, 'Age':'>40', 'Income':'Low', 'Student':'Yes', 'Credit_rating':'Excellent', 'Class: buys_computer':'No'},
        {'id':7, 'Age':'31-40', 'Income':'Low', 'Student':'Yes', 'Credit_rating':'Excellent', 'Class: buys_computer':'Yes'},
        {'id':8, 'Age':'<=30', 'Income':'Medium', 'Student':'No', 'Credit_rating':'Fair', 'Class: buys_computer':'No'},
        {'id':9, 'Age':'<=30', 'Income':'Low', 'Student':'Yes', 'Credit_rating':'Fair', 'Class: buys_computer':'Yes'},
        {'id':10, 'Age':'>40', 'Income':'Medium', 'Student':'Yes', 'Credit_rating':'Fair', 'Class: buys_computer':'Yes'},
        {'id':11, 'Age':'<=30', 'Income':'Medium', 'Student':'Yes', 'Credit_rating':'Excellent', 'Class: buys_computer':'Yes'},
        {'id':12, 'Age':'31-40', 'Income':'Medium', 'Student':'No', 'Credit_rating':'Excellent', 'Class: buys_computer':'Yes'},
        {'id':13, 'Age':'31-40', 'Income':'High', 'Student':'Yes', 'Credit_rating':'Fair', 'Class: buys_computer':'Yes'},
        {'id':14, 'Age':'>40', 'Income':'Medium', 'Student':'No', 'Credit_rating':'Excellent', 'Class: buys_computer':'No'}]

df = pd.DataFrame(data)
df
```

Data tersebut diatur dalam bentuk dictionary of lists dengan setiap kolom diwakili oleh key dalam dictionary dan setiap baris diwakili oleh sebuah list. Berikut merupakan penjelasan untuk masing-masing kolom dalam dataframe tersebut:

1) id: nomor identifikasi
2) Age: rentang usia pembeli (<=30, 31-40, atau >40)
3) Income: kisaran penghasilan pembeli (low, medium, atau high)
4) Student: apakah pembeli merupakan mahasiswa atau bukan (yes atau no)
5) Credit_rating: rating kredit pembeli (fair atau excellent)
6) Class: buys_computer: apakah pembeli direkomendasikan untuk membeli computer atau tidak (yes atau no)

Berikut merupakan output dari code di atas.

[2]:

	id	Age	Income	Student	Credit_rating	Class: buys_computer
0	1	<=30	High	No	Fair	No
1	2	<=30	High	No	Excellent	No
2	3	31-40	High	No	Fair	Yes
3	4	>40	Medium	No	Fair	Yes
4	5	>40	Low	Yes	Fair	Yes
5	6	>40	Low	Yes	Excellent	No
6	7	31-40	Low	Yes	Excellent	Yes
7	8	<=30	Medium	No	Fair	No
8	9	<=30	Low	Yes	Fair	Yes
9	10	>40	Medium	Yes	Fair	Yes
10	11	<=30	Medium	Yes	Excellent	Yes
11	12	31-40	Medium	No	Excellent	Yes
12	13	31-40	High	Yes	Fair	Yes
13	14	>40	Medium	No	Excellent	No

- Membuat dataframe baru yang masih sama dengan dataframe df, namun tanpa kolom id yang akan digunakan untuk mempermudah proses klasifikasi. Dataframe tersebut akan disimpan ke dalam variabel df2.

```
[3]: # membuat dataframe baru tanpa kolom id
     df2 = df.drop('id', axis=1)
```

1. **Perhitungan probabilitas tiap nilai nominal pada label kelas $[P(Ci)]$**
   - Secara manual, probabilitas tiap nilai nominal pada label kelas dapat dihitung menggunakan rumus:

$$P(Ci) = \frac{Jumlah\ data\ dengan\ kelas\ Ci}{Total\ jumlah\ data\ pada\ dataset}$$

   - Code script pada python.

```
[4]: for column in df2.columns:
         prob = df2[column].value_counts(normalize=True)
         print(f'\033[1mProbabilitas {column}:\033[0m\n{round(prob, 2)}')
         print()
```

   Code tersebut akan melakukan looping for untuk melakukan iterasi pada setiap kolom dalam dataframe df2. Pada setiap iterasi, akan dihitung probabilitas setiap nilai dalam kolom tersebut dengan menggunakan metode value_counts(normalize=True) yang akan disimpan ke dalam variabel prob. normalize=True akan menormalkan hasil sehingga menghasilkan nilai probabilitas dalam bentuk angka desimal antara 0 hingga 1.

   Kemudian, hasil probabilitas tersebut akan ditampilkan pada output menggunakan fungsi print(). \033[1m dan \033[0m digunakan untuk menambahkan format teks bold pada judul yang ditampilkan. round(prob, 2) digunakan untuk menampilkan nilai probabilitas bilangan desimal dengan membulatkan 2 angka di belakang koma.

   - Berikut merupakan output dari code di atas.

```
Probabilitas Age:
<=30     0.36
>40      0.36
31-40    0.29
Name: Age, dtype: float64

Probabilitas Income:
Medium    0.43
High      0.29
Low       0.29
Name: Income, dtype: float64

Probabilitas Student:
No     0.5
Yes    0.5
Name: Student, dtype: float64

Probabilitas Credit_rating:
Fair         0.57
Excellent    0.43
Name: Credit_rating, dtype: float64

Probabilitas Class: buys_computer:
Yes    0.64
No     0.36
Name: Class: buys_computer, dtype: float64
```

2. **Perhitungan probabilitas tiap nilai fitur/variabel prediktor (case) terhadap label kelas $[P(Xi|Ci)]$**
   - Secara manual, probabilitas tiap nilai fitur/variabel prediktor (case) terhadap label kelas dapat dihitung menggunakan rumus:

$$P(Xi|Ci) = \frac{Jumlah\ data\ pada\ kelas\ Ci\ dengan\ nilai\ fitur\ Xi}{Jumlah\ data\ pada\ kelas\ Ci}$$

- Code script pada python.

```
[5]: # menghitung probabilitas kondisional tiap nilai fitur/variabel prediktor untuk setiap kelas
     class_labels = df2['Class: buys_computer'].unique()
     for column in df2.columns[:-1]: # melakukan looping for kecuali kolom terakhir (kelas)
         values = df2[column].unique()
         print(f'\033[1mProbabilitas {column} terhadap Class: buys_computer:\033[0m')
         for value in values:
             for label in class_labels:
                 prob = len(df2[(df2[column] == value) & (df2['Class: buys_computer'] == label)]) / len(df2[df2['Class: buys_computer'] == label])
                 print(f'P({column} = "{value}" | buys_computer = "{label}") = {prob:.2f}')
         print()
```

df2[‘Class: buys_computer’].unique() digunakan untuk mendapatkan nilai unik dari kolom kelas yang akan disimpan ke dalam variabel class_labels. Selanjutnya, melakukan looping for untuk melakukan iterasi pada setiap kolom dalam dataframe df2 kecuali kolom terakhir (kelas). Pada setiap iterasi, akan mendapatkan nilai unik dari setiap kolom yang akan disimpan ke dalam variabel values. Lalu, nilai unik tersebut ditampilkan pada output dengan format teks bold menggunakan fungsi print().

Kemudian, melakukan looping for untuk melakukan iterasi setiap nilai pada kolom. Dilanjutkan melakukan looping for untuk setiap nilai kelas. Pada setiap iterasi, akan menghitung probabilitas dengan menghitung jumlah data nilai prediktor dan nilai unik dari kolom kelas. Setelah itu, dibagi dengan jumlah data nilai unik dari kolom kelas. Hasil probabilitas tersebut akan ditampilkan pada output menggunakan fungsi print(). (prob:.2)f digunakan untuk menampilkan nilai probabilitas bilangan desimal dengan membulatkan 2 angka di belakang koma.

- Berikut merupakan output dari code di atas.

```
Probabilitas Age terhadap Class: buys_computer:
P(Age = "<=30" | buys_computer = "No") = 0.60
P(Age = "<=30" | buys_computer = "Yes") = 0.22
P(Age = "31-40" | buys_computer = "No") = 0.00
P(Age = "31-40" | buys_computer = "Yes") = 0.44
P(Age = ">40" | buys_computer = "No") = 0.40
P(Age = ">40" | buys_computer = "Yes") = 0.33

Probabilitas Income terhadap Class: buys_computer:
P(Income = "High" | buys_computer = "No") = 0.40
P(Income = "High" | buys_computer = "Yes") = 0.22
P(Income = "Medium" | buys_computer = "No") = 0.40
P(Income = "Medium" | buys_computer = "Yes") = 0.44
P(Income = "Low" | buys_computer = "No") = 0.20
P(Income = "Low" | buys_computer = "Yes") = 0.33

Probabilitas Student terhadap Class: buys_computer:
P(Student = "No" | buys_computer = "No") = 0.80
P(Student = "No" | buys_computer = "Yes") = 0.33
P(Student = "Yes" | buys_computer = "No") = 0.20
P(Student = "Yes" | buys_computer = "Yes") = 0.67

Probabilitas Credit_rating terhadap Class: buys_computer:
P(Credit_rating = "Fair" | buys_computer = "No") = 0.40
P(Credit_rating = "Fair" | buys_computer = "Yes") = 0.67
P(Credit_rating = "Excellent" | buys_computer = "No") = 0.60
P(Credit_rating = "Excellent" | buys_computer = "Yes") = 0.33
```

3. **Pekalian tiap probabilitas nilai fitur/variabel prediktor terhadap label kelas dengan probabilitas tiap nilai nominal pada label kelas [$P(X|Ci)$]**
   - Secara manual, probabilitas tiap nilai fitur/variabel prediktor terhadap label kelas dengan probabilitas tiap nilai nominal pada label kelas dapat dihitung menggunakan rumus:

$$P(X|Ci) = P(X1|Ci) \times P(X2|Ci) \times \ldots \times P(Xn|Ci)$$

   - Code script pada python.

```
[6]: # menghitung probabilitas tiap nilai nominal pada label kelas
prob_age = df['Age'].value_counts(normalize=True)
prob_income = df['Income'].value_counts(normalize=True)
prob_student = df['Student'].value_counts(normalize=True)
prob_credit = df['Credit_rating'].value_counts(normalize=True)
prob_class = df['Class: buys_computer'].value_counts(normalize=True)

# membuat fungsi untuk menghitung probabilitas pada variabel/atribut/kasus dan kelas
def calculate_case_prob(row):
    prob_age_val = prob_age[row['Age']]
    prob_income_val = prob_income[row['Income']]
    prob_student_val = prob_student[row['Student']]
    prob_credit_val = prob_credit[row['Credit_rating']]
    prob_class_val = prob_class[row['Class: buys_computer']]
    return prob_age_val * prob_income_val * prob_student_val * prob_credit_val * prob_class_val

# menambahkan kolom baru case_prob
df['Case_prob'] = df.apply(calculate_case_prob, axis=1)
df
```

Pertama-tama, dilakukan perhitungan probabilitas setiap nilai dalam kolom dengan menggunakan fungsi value_counts dan normalizer=True seperti pada nomor 1. Hanya saja, pada code ini hasilnya akan disimpan ke dalam variabel prob_age, prob_income, prob_student, prob_credit, dan prob_class.

Kemudian, dibuat sebuah fungsi bernama calculate_case_prob() yang akan digunakan untuk menghitung probabilitas pada tiap variabel/atribut/kasus dan kelas yang selanjutnya seluruh hasil probabilitas tersebut akan dikalikan. Fungsi ini akan menerima sebuah argument row, yang berisi data pada tiap baris, kemudian mengambil nilai probabilitas pada tiap variabel/atribut/kasus dan kelas menggunakan variabel probabilitas yang sudah dihitung sebelumnya.

Terakhir, kolom baru bernama Case_prob ditambahkan pada dataframe df dengan menggunakan fungsi apply() dan argument axis=1 untuk melakukan operasi pada tiap baris. Fungsi yang digunakan adalah calculate_case_prob() yang sudah dibuat sebelumnya.

- Berikut merupakan output dari code di atas.

[6]:

	id	Age	Income	Student	Credit_rating	Class: buys_computer	Case_prob
0	1	<=30	High	No	Fair	No	0.010412
1	2	<=30	High	No	Excellent	No	0.007809
2	3	31-40	High	No	Fair	Yes	0.014994
3	4	>40	Medium	No	Fair	Yes	0.028113
4	5	>40	Low	Yes	Fair	Yes	0.018742
5	6	>40	Low	Yes	Excellent	No	0.007809
6	7	31-40	Low	Yes	Excellent	Yes	0.011245
7	8	<=30	Medium	No	Fair	No	0.015618
8	9	<=30	Low	Yes	Fair	Yes	0.018742
9	10	>40	Medium	Yes	Fair	Yes	0.028113
10	11	<=30	Medium	Yes	Excellent	Yes	0.021085
11	12	31-40	Medium	No	Excellent	Yes	0.016868
12	13	31-40	High	Yes	Fair	Yes	0.014994
13	14	>40	Medium	No	Excellent	No	0.011714

4. Membandingkan nilai probabilitas yang besar sebagai hasil prediksi label kelas dari record/instance

- Secara manual, membandingkan nilai probabilitas untuk mengetahui hasil prediksi label kelas dapat dihitung menggunakan rumus:

$$P(Ci|X) = P(X|Ci) \times P(Ci)$$

- Code script pada python.

```
# menghitung jumlah record pada dataset
total_records = len(df2)
```

Pertama-tama, menghitung jumlah record pada dataframe df2 yang akan disimpan ke dalam variabel total_records.

```
# menghitung jumlah record untuk label kelas
class_counts = df2['Class: buys_computer'].value_counts()
```

Selanjutnya, menghitung jumlah record untuk label kelas pada dataframe df2 menggunakan fungsi value_counts() yang akan disimpan ke dalam variabel class_counts.

```
# menghitung probabilitas prior untuk setiap kelas
priors = {}
for cls in class_counts.index:
    priors[cls] = class_counts[cls] / total_records
```

Kemudian, variabel priors akan menyimpan probabilitas prior untuk setiap kelas pada dataset. Lalu, akan dilakukan looping for untuk melakukan iterasi pada setiap index dalam class_counts. Pada setiap iterasi, probabilitas prior dihitung dengan membagi jumlah record kelas dengan total jumlah record.

```
# menghitung likelihood untuk setiap atribut pada setiap kelas
likelihoods = {}
for cls in class_counts.index:
    cls_data = df2[df2['Class: buys_computer'] == cls]
    cls_likelihoods = {}
    for col in df2.columns[:-1]:
        col_likelihoods = {}
        for val in cls_data[col].unique():
            col_likelihoods[val] = len(cls_data[cls_data[col] == val]) / class_counts[cls]
        cls_likelihoods[col] = col_likelihoods
    likelihoods[cls] = cls_likelihoods
```

Setelah itu, variabel likelihoods akan menyimpan likelihood untuk setiap atribut pada setiap kelas pada dataset. Likelihood merupakan probabilitas suatu nilai atribut diberikan label kelas tertentu pada dataset yang diberikan. Probabilitas ini dihitung dengan menghitung jumlah record dengan nilai tertentu dari atribut tersebut dibagi dengan jumlah record pada kelas tersebut.

```
# membuat fungsi untuk memprediksi kelas dari suatu instance
def predict(instance):
    posteriors = {}
    for cls in class_counts.index:
        likelihood = 1.0
        for col in df2.columns[:-1]:
            value = instance[col]
            if value in likelihoods[cls][col]:
                likelihood *= likelihoods[cls][col][value]
            else:
                likelihood = 0.0
                break
        posteriors[cls] = priors[cls] * likelihood
    return max(posteriors, key=posteriors.get)
```

Lalu, membuat fungsi predict yang menerima input berupa dictionary instance berisikan nilai-nilai atribut untuk sebuah instance. Fungsi ini menghitung probabilitas posterior untuk setiap kelas pada instance tersebut dengan mengalikan probabilitas prior dengan likelihood pada setiap atribut. Kombinasi probabilitas prior dan likelihood tersebut disimpan pada variabel posteriors. Fungsi max kemudian digunakan untuk mengembalikan kelas dengan probabilitas posteriors terbesar sebagai hasil prediksi

```
# mendapatkan prediksi untuk setiap baris dari hasil perbandingan probabilitas
for i in range(len(df2)):
    instance = df2.iloc[i]
    predicted_class = predict(instance)
    print('Predicted class for id', i+1, ':', predicted_class)
```

Terakhir, untuk mendapatkan prediksi dari setiap instance pada dataset akan dilakukan looping for dari baris pertama hingga baris terakhir pada dataframe df2. Kemudian, untuk setiap baris dilakukan prediksi kelas menggunakan fungsi predict yang sudah didefinisikan sebelumnya. Hasil prediksi kemudian akan ditampilkan pada output beserta dengan id barisnya.

- Berikut merupakan output dari code di atas.

```
Predicted class for id 1 : No
Predicted class for id 2 : No
Predicted class for id 3 : Yes
Predicted class for id 4 : Yes
Predicted class for id 5 : Yes
Predicted class for id 6 : Yes
Predicted class for id 7 : Yes
Predicted class for id 8 : No
Predicted class for id 9 : Yes
Predicted class for id 10 : Yes
Predicted class for id 11 : Yes
Predicted class for id 12 : Yes
Predicted class for id 13 : Yes
Predicted class for id 14 : No
```

5. **Pembagian data training dan data testing menggunakan percentage split (missal: 70:30, 80:20, dan 90:10)**
   - Code script pada python.

```
[9]: import random

     # Shuffle dataset
     df = df.sample(frac=1, random_state=42)

     # Pehitungan persentasi data training dan data testing
     train_size = int(0.7 * len(df))

     # Split data ke dalam data training and data testing
     train_data = df[:train_size]
     test_data = df[train_size:]

     print("\033[1mTraining data:\n\033[0m", train_data)
     print()
     print("\033[1mTesting data:\n\033[0m", test_data)
```

   Pertama-tama import random terlebih dahulu. Lalu, data pada dataframe df diacak menggunakan fungsi sample() dengan parameter frac=1 (menandakan seluruh data akan diambil) dan random_state=42 (sebagai seed untuk memastikan pengacakan yang sama setiap kali code dijalankan). Pada code ini akan dilakukan pembagian data training sebesar 70% dan data testing sebesar 30%. Maka dari itu, variabel train_size akan menyimpan perkalian jumlah data total dengan 0.7, lalu dibulatkan ke bawah ke integer terdekat dengan menggunakan fungsi int(). Data pada dataframe df kemudian dibagi menjadi dua, yaitu data training yang berisi data dari indeks ke-0 hingga indeks ke-(train_size-1) dan data testing yang berisi data dari indeks ke-(train_size) hingga indeks terakhir. Data training dan data testing akan ditampilkan pada output menggunakan fungsi print().

   - Berikut merupakan output dari code di atas.

```
Training data:
     id    Age  Income Student Credit_rating Class: buys_computer  Case_prob
9    10    >40  Medium     Yes          Fair                  Yes   0.028113
11   12  31-40  Medium      No     Excellent                  Yes   0.016868
0     1   <=30    High      No          Fair                   No   0.010412
12   13  31-40    High     Yes          Fair                  Yes   0.014994
5     6    >40     Low     Yes     Excellent                   No   0.007809
8     9   <=30     Low     Yes          Fair                  Yes   0.018742
2     3  31-40    High      No          Fair                  Yes   0.014994
1     2   <=30    High      No     Excellent                   No   0.007809
13   14    >40  Medium      No     Excellent                   No   0.011714

Testing data:
     id    Age  Income Student Credit_rating Class: buys_computer  Case_prob
4     5    >40     Low     Yes          Fair                  Yes   0.018742
7     8   <=30  Medium      No          Fair                   No   0.015618
10   11   <=30  Medium     Yes     Excellent                  Yes   0.021085
3     4    >40  Medium      No          Fair                  Yes   0.028113
6     7  31-40     Low     Yes     Excellent                  Yes   0.011245
```

6. **Perhitungan performa klasifikasi menggunakan confussion matrix (Accuracy, Recall, Precision, F-measure)**
   - Code script pada python.

```python
# membuat fungsi untuk menghitung performance matrix
def calculate_matrix(tp, tn, fp, fn):
    accuracy = (tp + tn) / (tp + tn + fp + fn)
    recall = tp / (tp + fn)
    precision = tp / (tp + fp)
    f_measure = 2 * precision * recall / (precision + recall)
    return accuracy, recall, precision, f_measure
```

Fungsi calculate_matrix(tp, tn, fp, fn) digunakan untuk menghitung performance matrix (akurasi, recall, precision, f-measure) dari confusion matrix dengan parameter tp (true positive), tn (true negative), fp (false positive), dan fn (false negative).

```python
# membuat fungsi untuk menghitung confusion matrix
def calculate_confusion_matrix(predictions, actuals):
    tp = np.sum((predictions == 'Yes') & (actuals == 'Yes'))
    tn = np.sum((predictions == 'No') & (actuals == 'No'))
    fp = np.sum((predictions == 'Yes') & (actuals == 'No'))
    fn = np.sum((predictions == 'No') & (actuals == 'Yes'))
    return tp, tn, fp, fn
```

Fungsi calculate_confusion_matrix(predictions, actuals) digunakan untuk menghitung confusion matrix dengan parameter predictions (array berisi hasil prediksi dari model) dan actuals (array berisi label asli dari data testing).

```python
# Prediksi variabel target menggunakan kelas mayoritas pada training set
majority_class = train_data['Class: buys_computer'].value_counts().index[0]
predictions = np.array([majority_class] * len(test_data))
actuals = np.array(test_data['Class: buys_computer'])

# Menghitung confusion matrix dan performance matrix
tp, tn, fp, fn = calculate_confusion_matrix(predictions, actuals)
accuracy, recall, precision, f_measure = calculate_matrix(tp, tn, fp, fn)

# Print performance matrix
print("Accuracy:", accuracy)
print("Recall:", recall)
print("Precision:", precision)
print("F-measure:", f_measure)
```

Pada baris selanjutnya, dilakukan prediksi variabel target menggunakan kelas mayoritas pada data training. Setelah itu, dilakukan perhitungan confusion matrix dan performance matrix menggunakan fungsi calculate_confusion_matrix() dan calculate_matrix(). Hasilnya disimpan pada variabel tp, tn, fp, fn, accuracy, recall, precision, dan f_measure. Hasil performance matrix kemudian dicetak pada layar menggunakan fungsi print().

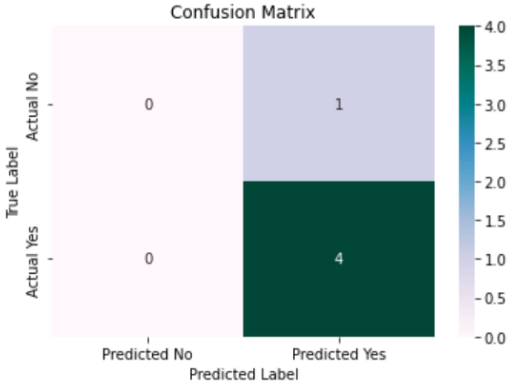
```python
# Plot confusion matrix
confusion_matrix = pd.DataFrame([[tn, fp], [fn, tp]], index=['Actual No', 'Actual Yes'], columns=['Predicted No', 'Predicted Yes'])
sns.heatmap(confusion_matrix, annot=True, cmap='PuBuGn', fmt='g')

# Berikan judul plot dan label axis
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

Selanjutnya, confusion matrix diplot menggunakan library seaborn dengan fungsi sns.heatmap(). Plot confusion matrix ditampilkan pada layar menggunakan fungsi plt.show().

- Berikut merupakan output dari code di atas.

```
Accuracy: 0.8
Recall: 1.0
Precision: 0.8
F-measure: 0.888888888888889
```



Berdasarkan hasil prediksi model di atas, dapat dilakukan evaluasi performa klasifikasi dengan menggunakan beberapa metrik evaluasi performa klasifikasi, yaitu:

- Akurasi (Accuracy): 0.8
  Akurasi mengukur seberapa banyak prediksi yang benar dibandingkan dengan jumlah total prediksi yang dilakukan. Dalam kasus ini, nilai akurasi model adalah 0.8, yang berarti 80% dari prediksi yang dilakukan oleh model adalah benar.
- Recall (Sensitivitas): 1.0
  Recall mengukur seberapa banyak prediksi positif yang benar dari total jumlah data aktual yang positif. Dalam kasus ini, nilai recall model adalah 1.0, yang berarti model dapat memprediksi semua data aktual yang positif dengan benar.
- Presisi (Precision): 0.8
  Presisi mengukur seberapa banyak prediksi positif yang benar dari total prediksi positif. Dalam kasus ini, nilai presisi model adalah 0.8, yang berarti 80% dari prediksi positif model adalah benar.
- F1-Score: 0.888888888888889
  F1-Score mengukur rata-rata harmonik antara presisi dan recall. Dalam kasus ini, nilai F1-Score model adalah 0.888888888888889, yang menunjukkan bahwa model memiliki keseimbangan yang baik antara presisi dan recall.

Dari evaluasi performa klasifikasi tersebut, dapat disimpulkan bahwa model memiliki nilai akurasi yang baik, namun memiliki nilai presisi yang kurang optimal. Hal ini menunjukkan bahwa model cenderung memberikan banyak prediksi positif palsu. Selain itu, model memiliki nilai recall dan F1-Score yang sangat baik, yang menunjukkan bahwa model dapat memprediksi semua data aktual yang positif dengan benar.